Ministério da Educação
Universidade Tecnológica Federal do Paraná
Câmpus Apucarana
Bacharelado em Engenharia de Computação

Universidade Tecnológica Federal do Paraná

Engenharia de Computação

IMPLEMENTAÇÃO DE UM COFRE UTILIZANDO SENSORES E ARDUINO

Maria Eduarda Pedroso

Vítor Augusto Ozanick

Professor orientador: Mauricio Nakai

Apucarana

Dezembro/2023

SUMÁRIO

1. RESUMO

Este trabalho apresenta o desenvolvimento de um Sistema de Controle de Acesso baseado no Arduino Mega 2560. Integrando componentes como RFID, teclado, servo motor, LCD e sensor magnético, o sistema oferece autenticação segura por meio de RFID e senha. O projeto visa criar uma solução robusta e expansível, aproveitando as capacidades do Arduino Mega 2560. Feedback visual e sonoro são proporcionados através de um display LCD e um buzzer, comunicando mensagens relevantes durante o processo de autenticação. Medidas de segurança, incluindo bloqueio temporário após tentativas incorretas e "torturas eletrônicas", são implementadas para fortalecer a proteção contra acessos não autorizados. Destaca-se ainda o uso eficiente do Arduino Mega 2560, evidenciado pela otimização da conectividade e expansibilidade do sistema. O trabalho contribui para a compreensão prática das habilidades de programação e integração de hardware adquiridas ao longo do curso.

Palavras-chave: Controle de Acesso, Arduino Mega 2560, RFID, Teclado, Segurança, Sistema Embarcado.

2. INTRODUÇÃO

Este relatório descreve o desenvolvimento de um sistema de segurança com controle de acesso baseado em um microcontrolador. O projeto visa criar um mecanismo seguro de acesso a espaços restritos, utilizando elementos como teclado, RFID, eletroímã e um servo motor para controle físico da tranca.

O sistema é projetado para gerenciar o acesso a um ambiente restrito, incorporando a autenticação por meio de senha numérica inserida via teclado e a utilização de cartões RFID para permitir a alteração da senha.

Ao longo deste relatório, serão apresentados os componentes utilizados, a lógica de funcionamento do sistema, detalhes de implementação do código e a interação entre os diferentes módulos para garantir o controle de acesso eficiente e seguro.

O código desenvolvido para este projeto abrange desde a configuração dos componentes até a implementação das funcionalidades de verificação de senhas, leitura de cartões RFID, controle do eletroímã e servo motor, além de mecanismos de segurança para lidar com possíveis tentativas de acesso não autorizadas.

Agora dando ênfase no microcontrolador escolhido, o Arduino é uma plataforma de prototipagem eletrônica amplamente utilizada, oferece um ambiente acessível e flexível para implementação de sistemas de controle. Equipado com uma variedade de pinos de entrada e saída, o Arduino é capaz de interagir com sensores, atuadores e outros dispositivos periféricos.

No contexto do sistema de controle de acesso, a implementação do Arduino se destaca ao oferecer flexibilidade na gestão de múltiplos métodos de autenticação, como cartões e chaveiros RFID, combinados com a inserção de senhas numéricas. O Arduino coordena as interações entre esses métodos, processando entradas do

usuário por meio do teclado e verificando a autenticidade dos cartões RFID para controle de acesso.

A utilização de bibliotecas, como a MFRC522 para RFID e a Keypad para o teclado, simplifica a integração desses dispositivos ao sistema, aumentando a eficiência e a modularidade do código.

2.1. Objetivos

O presente trabalho tem como objetivo:

- Desenvolver um Sistema de Controle de Acesso: Criar um sistema que permita controlar o acesso a uma porta, combinando métodos de autenticação por RFID e senha.
- Integração de Componentes Eletrônicos: Integrar diversos componentes eletrônicos, como RFID, teclado, servo motor, LCD e sensor magnético, para criar um sistema coeso e funcional.
- Proporcionar Segurança no Controle de Acesso: Garantir que apenas usuários autorizados possam acessar o sistema, implementando medidas de segurança contra tentativas não autorizadas.

2.2. Materiais e equipamentos

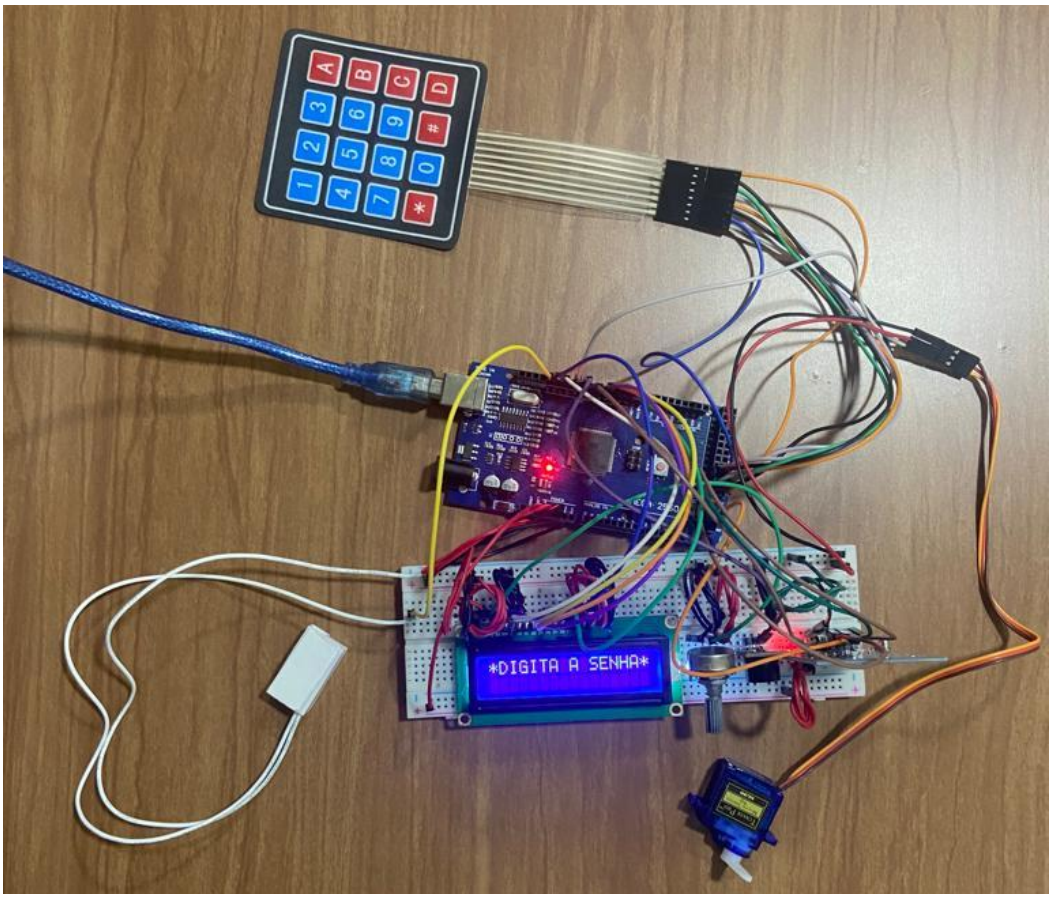
- Arduino Mega 2560: O Arduino Mega 2560 é uma placa de desenvolvimento baseada no chip ATmega2560. Ele é uma versão avançada do Arduino Uno, oferecendo maior capacidade de processamento, mais entradas/saídas digitais e analógicas, além de mais recursos de memória. Algumas características principais incluem:

- RFID Module (MFRC522): O módulo RFID (MFRC522) é utilizado para ler informações de cartões RFID. Ele possui uma antena embutida e se comunica com o Arduino através do protocolo SPI (Serial Peripheral Interface).
- Keypad: O teclado (keypad) é uma matriz de botões utilizado para inserir a senha no sistema. Ele permite a entrada de dados de forma conveniente.
- Servo Motor: Um motor servo é utilizado para controlar o mecanismo de travamento da porta. Ele pode ser girado para uma posição específica, permitindo abrir ou fechar a porta.
- LCD Display (16x2): O display LCD (16x2) fornece feedback visual ao usuário, exibindo mensagens como "Digite a Senha" ou "Acesso Concedido".
- Buzzer: O buzzer é um dispositivo que emite sons. No seu projeto, ele é usado para emitir beeps durante a entrada de senha e também para reproduzir uma melodia durante a inicialização.
- Magnetic Sensor: Um sensor magnético é utilizado para detectar o estado da porta (aberta ou fechada). Ele fornece feedback sobre o status físico da porta.
- Jumpers: Jumpers são cabos condutores utilizados para fazer conexões entre os componentes. Eles possuem conectores em ambas as extremidades, facilitando a montagem do circuito.
- Potenciômetro: Um potenciômetro é um dispositivo variável que pode ser ajustado para fornecer uma resistência variável. No contexto do projeto, foi utilizado ajustar o brilho do LCD.

3. METODOLOGIA

O presente trabalho foi desenvolvido utilizando os conceitos e técnicas estudados durante as aulas da disciplina. Bem como foi empregado o uso de componentes eletrônicos unidos ao microprocessador arduino e suas bibliotecas, de modo a facilitar e simular nossa lógica, abaixo temos uma imagem da montagem do circuito

Figura 1 - Circuito montado.

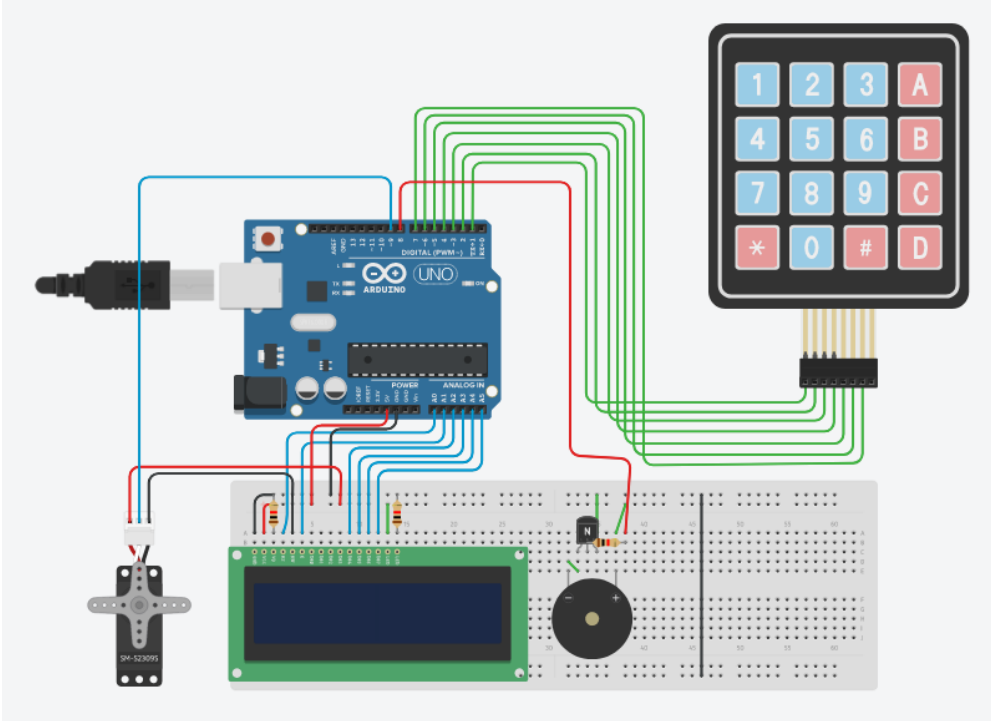


Fonte: Autoria Própria

3.1. Projeto inicial no tinkerCad

O projeto inicial no Tinkercad consistia em um sistema de controle de acesso simulado com Arduino, teclado, LCD, servo motor e buzzer, como mostra a figura abaixo.

Figura 2 - Circuito inicial.



Fonte: Autoria Própria

Posteriormente, foi incrementado com a adição de um módulo RFID e um sensor magnético de tranca, que simula a abertura e o fechamento da porta controlada pelo servo motor. O Arduino gerencia a entrada de senha via teclado, exibe informações no LCD, controla o servo motor para simular o acesso e fornece feedback sonoro pelo buzzer.

3.2. Modelagem 3D da porta

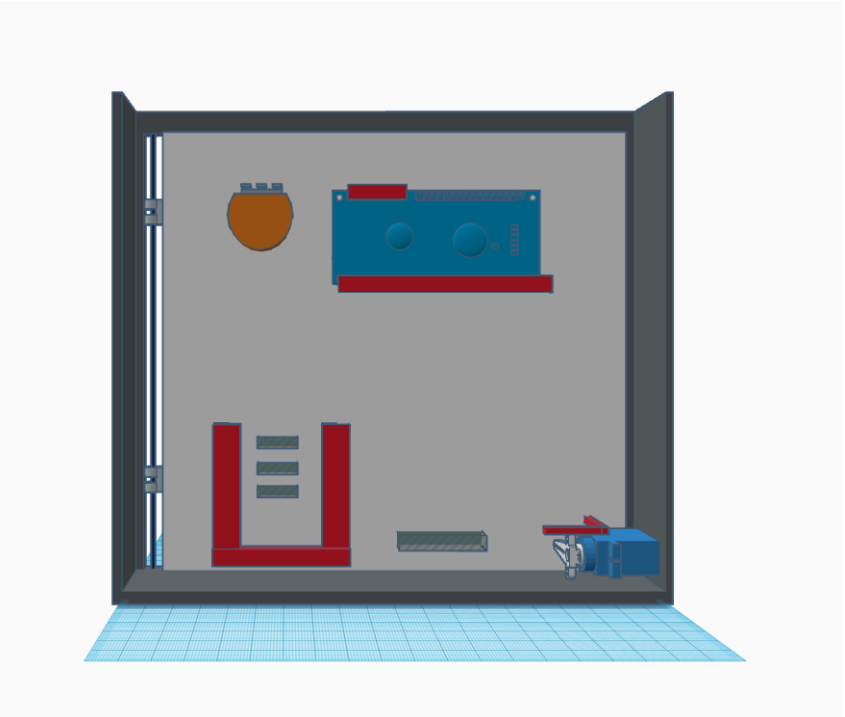
Durante a montagem do circuito e o desenvolvimento do código, simultaneamente elaboramos um modelo no TinkerCad representando a estrutura de uma porta de cofre. Esse modelo proporciona uma visualização clara e detalhada da disposição dos componentes no contexto do projeto como é apresentado a seguir.

Figura 3 - Visão frontal porta.

Fonte: Autoria Própria

Através da figura 2, observamos a visão frontal da porta do cofre, onde são visíveis o teclado para inserção da senha, a tela de LCD e orifícios destinados à passagem da membrana do teclado na parte inferior da porta. Ao lado, encontramos a área de aproximação do RFID, enquanto na parte superior está disposto o potenciômetro responsável pelo controle da luminosidade na tela de LCD.

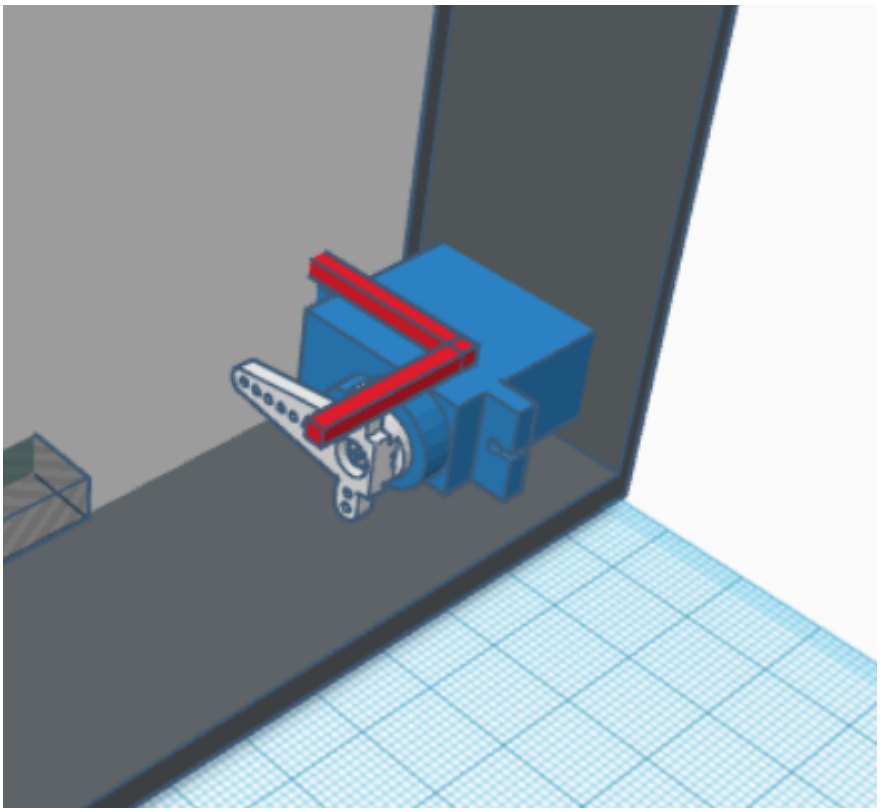
Figura 4 - Visão posterior porta.



Fonte: Autoria Própria

Na parte traseira da porta, foram modeladas estruturas para servir como suporte aos componentes. Na seção superior, há o suporte para o multímetro e o LCD, enquanto na porção inferior, o suporte é destinado ao sensor de RFID.

Figura 5 - Visão detalhada da trava de segurança.



Fonte: Autoria Própria

Na parte traseira, foi montada uma estrutura para travar a porta antes da liberação via senha. Nessa estrutura, o servo-motor atua como trava, impedindo a abertura do cofre.

3.3. Funcionalidades do projeto

3.3.1. Bibliotecas

Foram utilizadas as bibliotecas Keypad, LiquidCrystal, Servo, SPI e MFRC522 para o desenvolvimento deste projeto. A biblioteca Keypad é empregada para lidar com teclados matriciais, a LiquidCrystal para controlar displays LCD, a Servo para

controlar servo motores, a SPI para comunicação e a MFRC522 para utilização de módulos RFID. Abaixo temos a inicialização das bibliotecas.

```C/C++
#include <Keypad.h>
#include <LiquidCrystal.h>
#include <Servo.h>
#include <SPI.h>
#include <MFRC522.h>
```

No ambiente de desenvolvimento do Arduino, a instalação das bibliotecas é realizada por meio da plataforma Arduino IDE, acessando o menu "Sketch" e selecionando "Incluir Biblioteca" > "Gerenciar Bibliotecas". Em seguida, busca-se a biblioteca e instala a versão mais recente disponível.

3.3.2. Configurações Iniciais

Como todo projeto de software precisamos começar conectando pinos, definindo entradas e saídas e iniciando variáveis. Os pinos utilizados para conectar os componentes, como o buzzer, teclado matricial, display LCD, servo motor, módulo RFID e sensor magnético de tranca devem ser declarados e configurados. Abaixo vemos essa configuração.

```C/C++
// Configuração do buzzer

#define NOTE_E5 659
#define NOTE_B4 494
```

```
#define NOTE_D5 587
#define NOTE_C5 523

// Configuração do módulo RFID
#define RST_PIN 7
#define SS_PIN 53
MFRC522 mfrc522(SS_PIN, RST_PIN);

// Configuração do sensor magnético da tranca
const int SENSOR_PIN = 13;
const int DARK_THRESHOLD = 100;

// Configuração do servo motor
Servo myservo;
const int redled = 10;
const int greenled = 11;
const int buzz = 8;
int pos = 0;

// Configuração do LCD
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);
const byte rows = 4;
const byte cols = 3;

// Configuração do teclado
const byte qtdLinhas = 4;
const byte qtdColunas = 4;
char matriz_teclas[qtdLinhas][qtdColunas] = {
 {'1', '2', '3', 'A'},
 {'4', '5', '6', 'B'},
 {'7', '8', '9', 'C'},
```

```
 {'*', '0', '#', 'D'}
};
byte PinosqtdLinhas[qtdLinhas] = {38, 39, 40, 41};
byte PinosqtdColunas[qtdColunas] = {42, 43, 44, 45};
Keypad meuteclado = Keypad(makeKeymap(matriz_teclas),
PinosqtdLinhas, PinosqtdColunas, qtdLinhas, qtdColunas);
```

Essas etapas de configuração garantem que os dispositivos estejam prontos para operar e permitam a execução adequada do código principal no loop do programa.

3.3.3. Inicialização dos dispositivos

A função ‘setup()’ configura e inicializa os dispositivos usados no programa, além de tocar uma melodia inicial para indicar que o sistema está pronto.

C/C++

```
// Função de inicialização
void setup() {
 displayscreen(); // Inicializa o LCD
 Serial.begin(9600); // Inicializa a comunicação serial
  pinMode(redled, OUTPUT); // Configura o LED vermelho
como saída
 pinMode(greenled, OUTPUT); // Configura o LED verde como
saída
 pinMode(buzz, OUTPUT); // Configura o buzzer como saída
 myservo.attach(9); // Anexa o servo motor
  pinMode(SENSOR_PIN, INPUT_PULLUP); // Configura o pino
do sensor magnético
```

```
 lcd.begin(16, 2); // Inicializa o LCD

 while (!Serial);    // Aguarda a inicialização da porta serial
 SPI.begin();  // Inicializa o barramento SPI
 mfrc522.PCD_Init();  // Inicializa o módulo RFID
 delay(4);     // Atraso opcional após a inicialização
 mfrc522.PCD_DumpVersionToSerial(); // Exibe detalhes do módulo RFID
  Serial.println(F("Scan PICC to see UID, SAK, type, and data blocks...")); // Mensagem para escanear PICC

 inicialTune(); // Toca uma melodia inicial
}
```

3.3.4. Funções auxiliares

Existem várias funções auxiliares que realizam tarefas específicas, como trocar a tag RFID, trocar a senha, controlar o LCD, tocar melodias e manipular o servo motor para abrir e fechar a porta, dentre elas estão. Irei apresentar algumas dessas funções a seguir:

1. displayscreen(): Essa função é responsável por inicializar e limpar o display LCD. É usada para exibir a tela inicial, indicando onde o usuário deve digitar a senha.
2. keypress(): Esta função emite um som curto sempre que uma tecla é pressionada no teclado matricial, oferecendo um feedback sonoro para cada entrada do usuário.

3. incorrect(): Chamada quando a senha inserida pelo usuário está incorreta. Esta função exibe uma mensagem no LCD, acende um LED vermelho e emite um som de alerta usando o buzzer.
4. counterbeep(): Essa função gera uma sequência de beeps para indicar que a porta está prestes a se fechar. Ela conta para trás no display LCD e emite um som, alertando sobre o fechamento iminente.
5. unlockdoor(): Quando a senha correta é inserida ou a tag RFID é verificada corretamente, essa função é chamada. Ela abre a porta (simbolizada pelo servo motor), acende um LED verde, emite um som indicando a abertura e atualiza a tela do LCD para mostrar uma mensagem de boas-vindas.
6. inicialTune(): Toca uma melodia inicial ao ligar o dispositivo para indicar que está pronto para ser usado.
7. armservo(): Esta função controla a posição do servo motor, movendo-o para uma posição específica, possivelmente usada para algum propósito adicional no projeto.
8. unlockbuzz(): Emite um som de desbloqueio através do buzzer quando a porta é aberta com sucesso.
9. torture1() e torture2(): Duas funções que representam "torturas" caso uma senha incorreta seja inserida repetidamente. Elas exibem mensagens e emitem sons, simbolizando um aviso ou tempo de espera antes de permitir novas tentativas.

3.3.5. Função principal

No início do loop(), é verificado se o sistema está no modo de administrador. Se estiver, o código aguarda uma entrada para escolher entre a troca da tag ou da senha. Depois disso, se a porta estiver aberta para o usuário, exibe uma mensagem de boas-vindas no LCD.

Se não estiver no modo administrador, o código verifica a presença de um cartão RFID. Se um cartão é detectado, verifica se é um cartão autorizado e, se for, muda para o modo administrador. Caso contrário, permanece no modo normal.

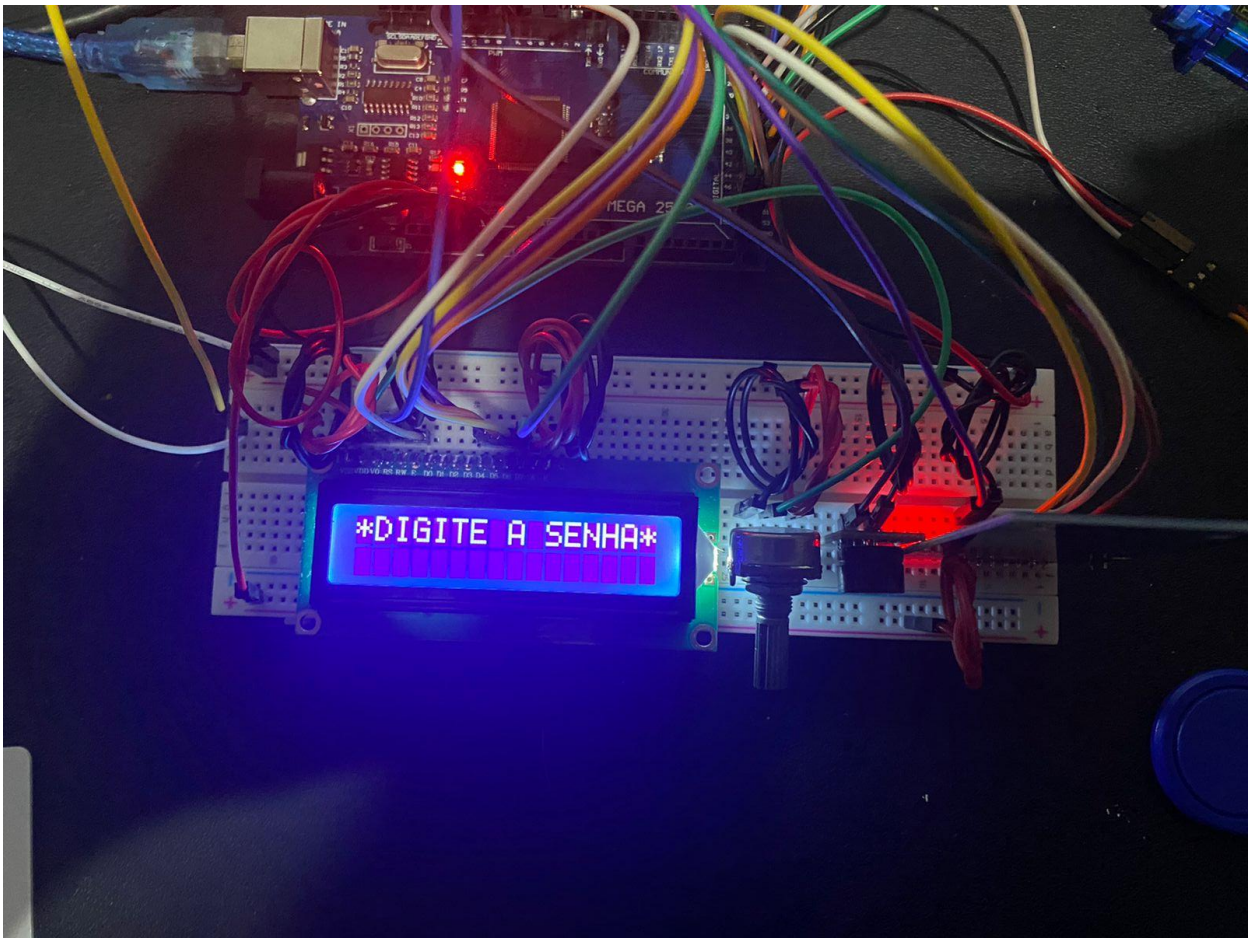
Se não estiver lendo um cartão RFID e a porta estiver fechada, o código aguarda a entrada do usuário pelo teclado matricial para inserir uma senha. A cada tecla pressionada, a senha é verificada e, se estiver correta, a porta é aberta. Caso contrário, há uma contagem de tentativas inválidas. Se houver muitas tentativas, o sistema aciona sequências de ‘tortura’ até que a senha correta seja inserida ou haja troca via RFID.

Esse processo de verificação e reação contínua é o que permite ao sistema funcionar de maneira autônoma e responder de forma adequada às interações do usuário.

4. RESULTADOS E DISCUSSÃO

Os resultados obtidos destacam a eficiência e segurança do sistema de controle de acesso ao cofre. Durante os testes, o sistema demonstrou sua capacidade de validar corretamente os cartões RFID autorizados, assim como processar e verificar as senhas digitadas. A robustez do sistema também se mostrou evidente ao lidar com tentativas inválidas de acesso. Quando um número predefinido de tentativas incorretas era alcançado, o sistema ativava sequências de tortura, reforçando a proteção e prevenindo acessos não autorizados. Apresentamos abaixo uma sequência de imagens ilustrativas que demonstram as operações realizadas pelo sistema desenvolvido.

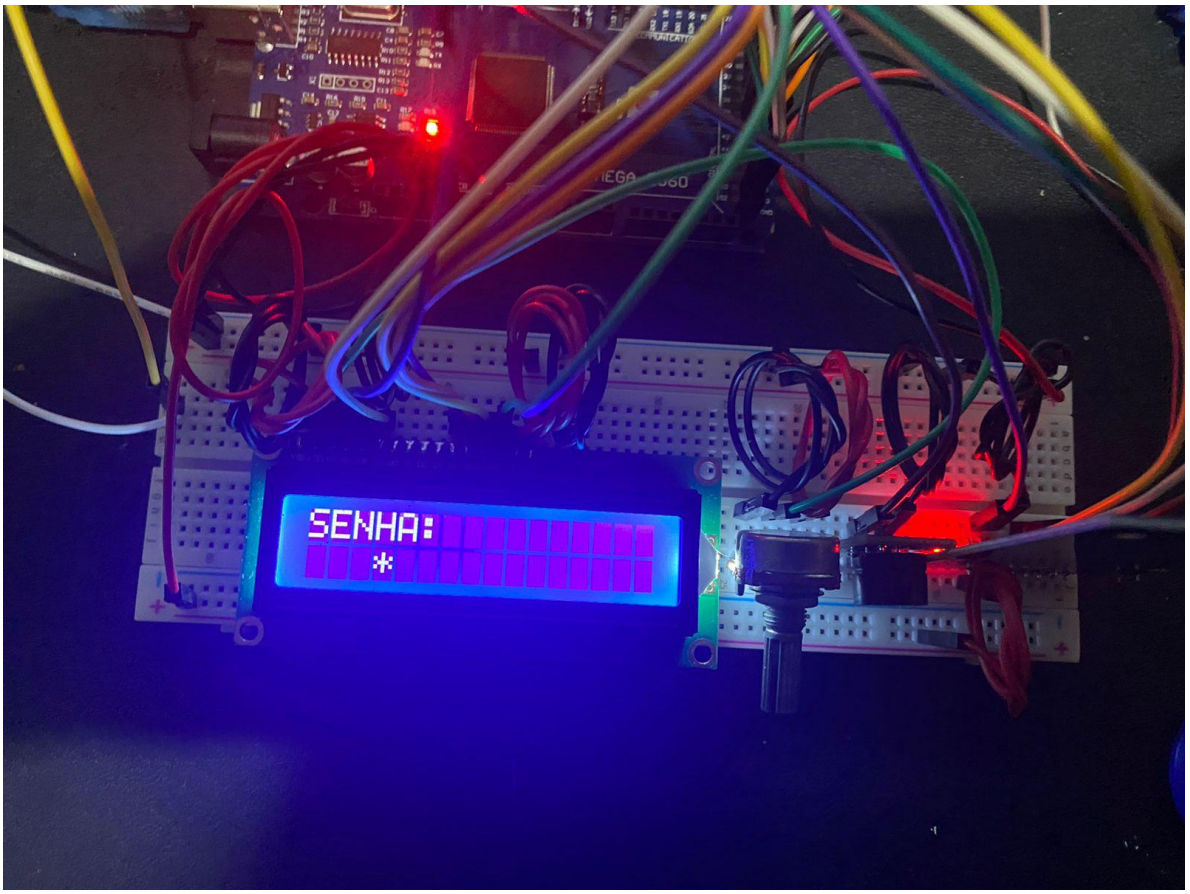
Figura 6 - Solicita senha



Fonte: Autoria Própria

Ao acionar o sistema, uma tela de solicitação de senha é apresentada no display LCD.

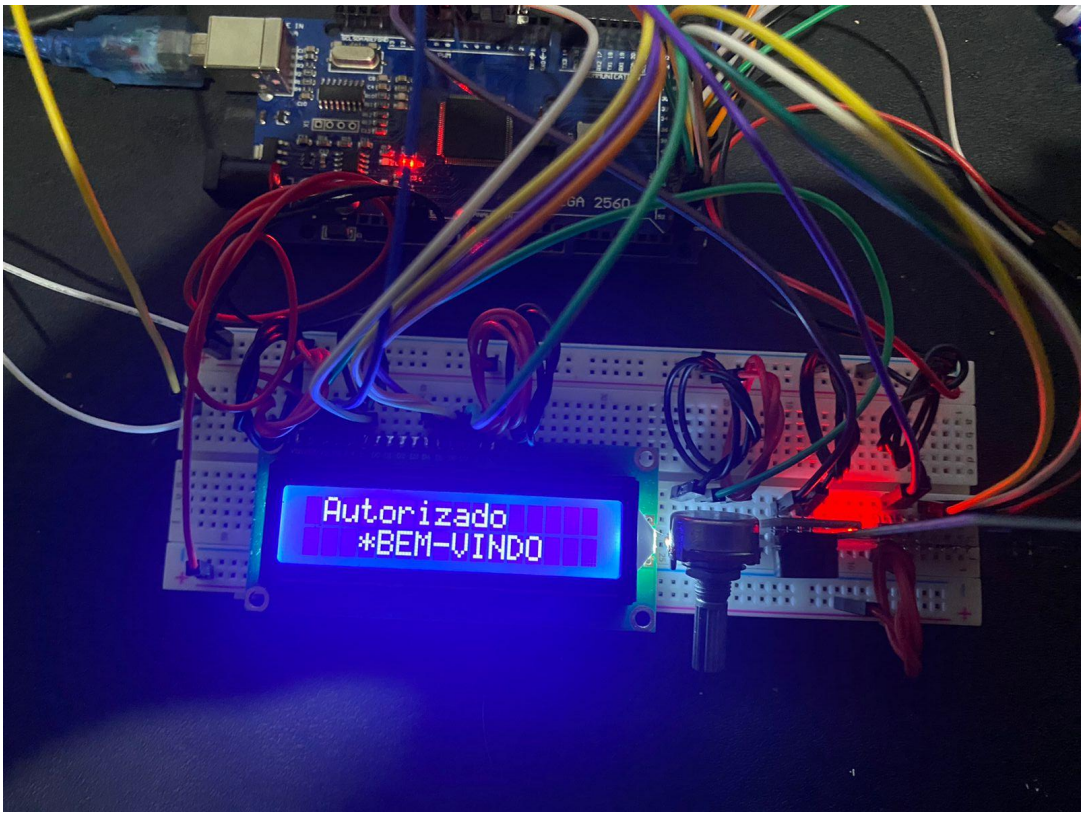
Figura 7 - Aquisição da senha



Fonte: Autoria Própria

Durante o processo de inserção da senha, os dígitos digitados são ocultados no display LCD.

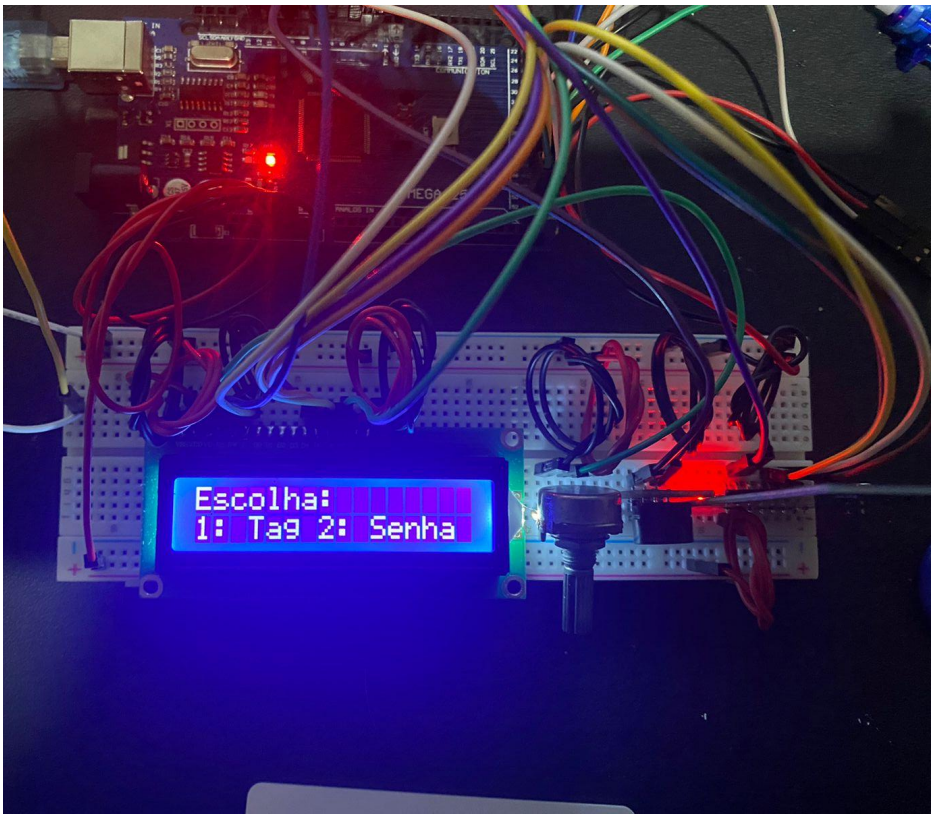
Figura 8 - Resultado de senha correta.



Fonte: Autoria Própria

Quando a senha correta é inserida, o cofre é desbloqueado e uma mensagem de boas-vindas é exibida.

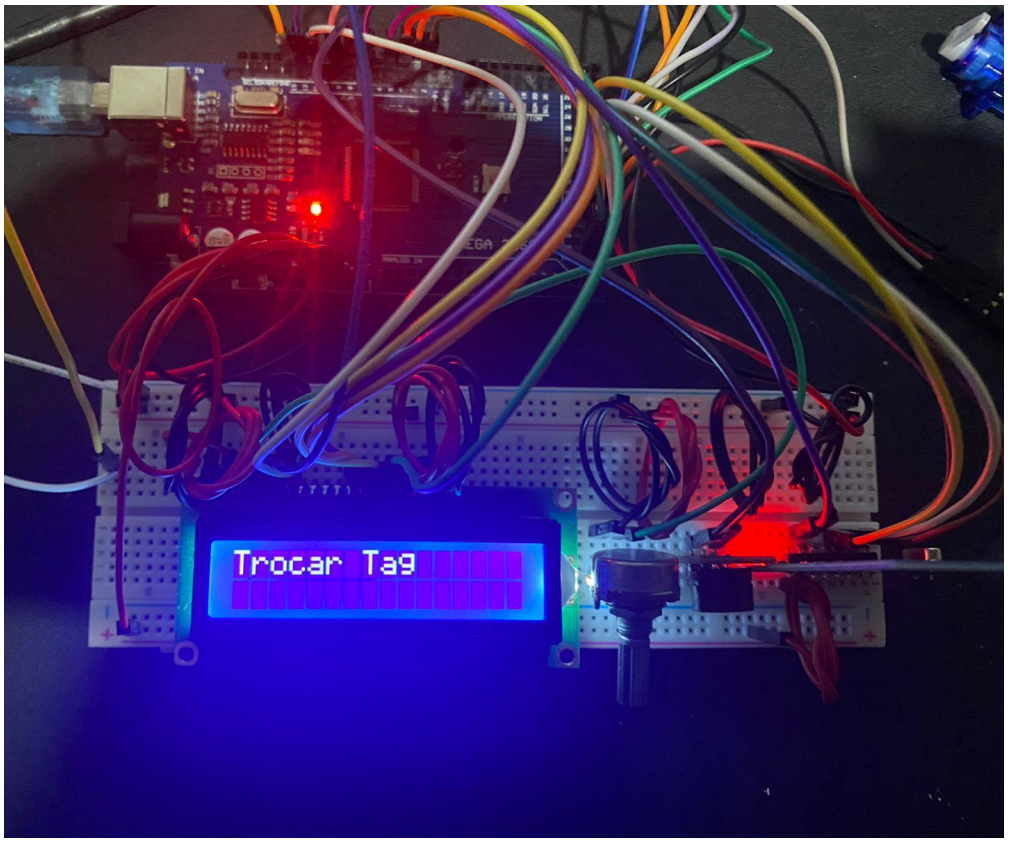
Figura 9 - Troca senha ou tag.



Fonte: Autoria Própria

Ao aproximar o cartão de identificação, o sistema oferece opções permitindo a escolha entre a alteração da senha ou a substituição da tag de identificação.

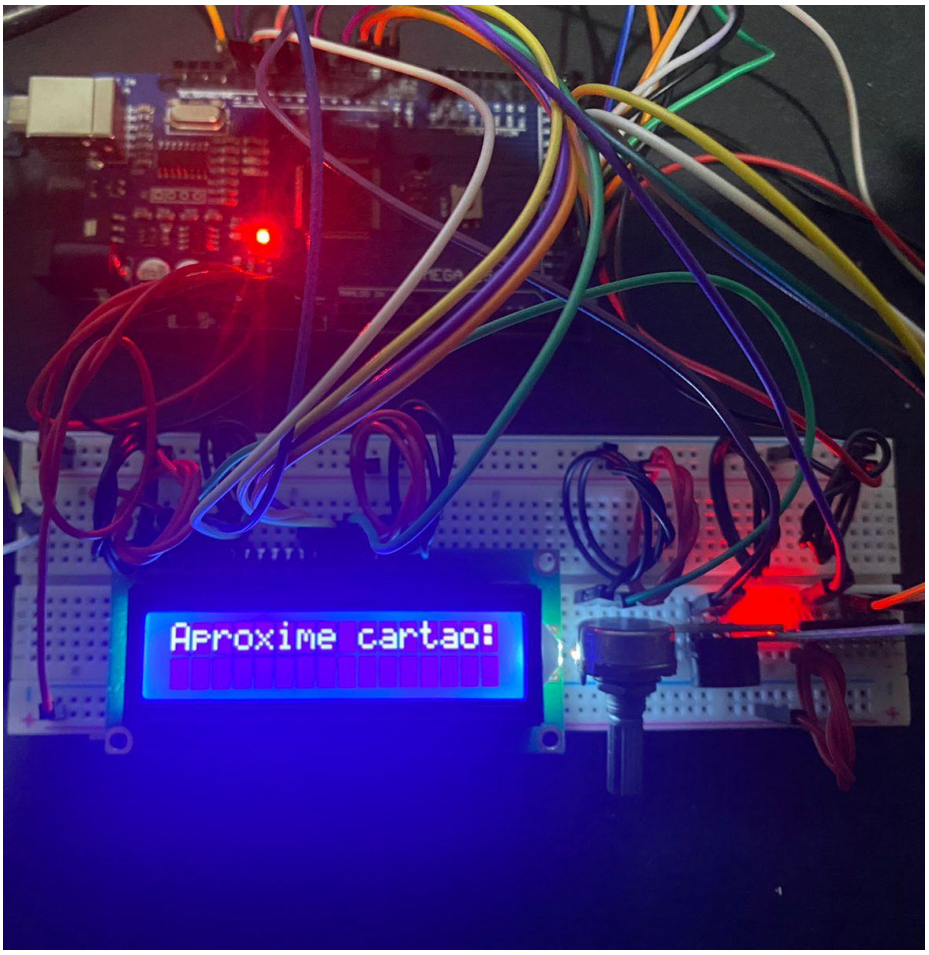
Figura 10 - Troca de tag



Fonte: Autoria Própria

Quando a opção de trocar a tag é selecionada, o display exibe essa escolha e solicita ao usuário que aproxime o chaveiro RFID para efetuar a substituição.

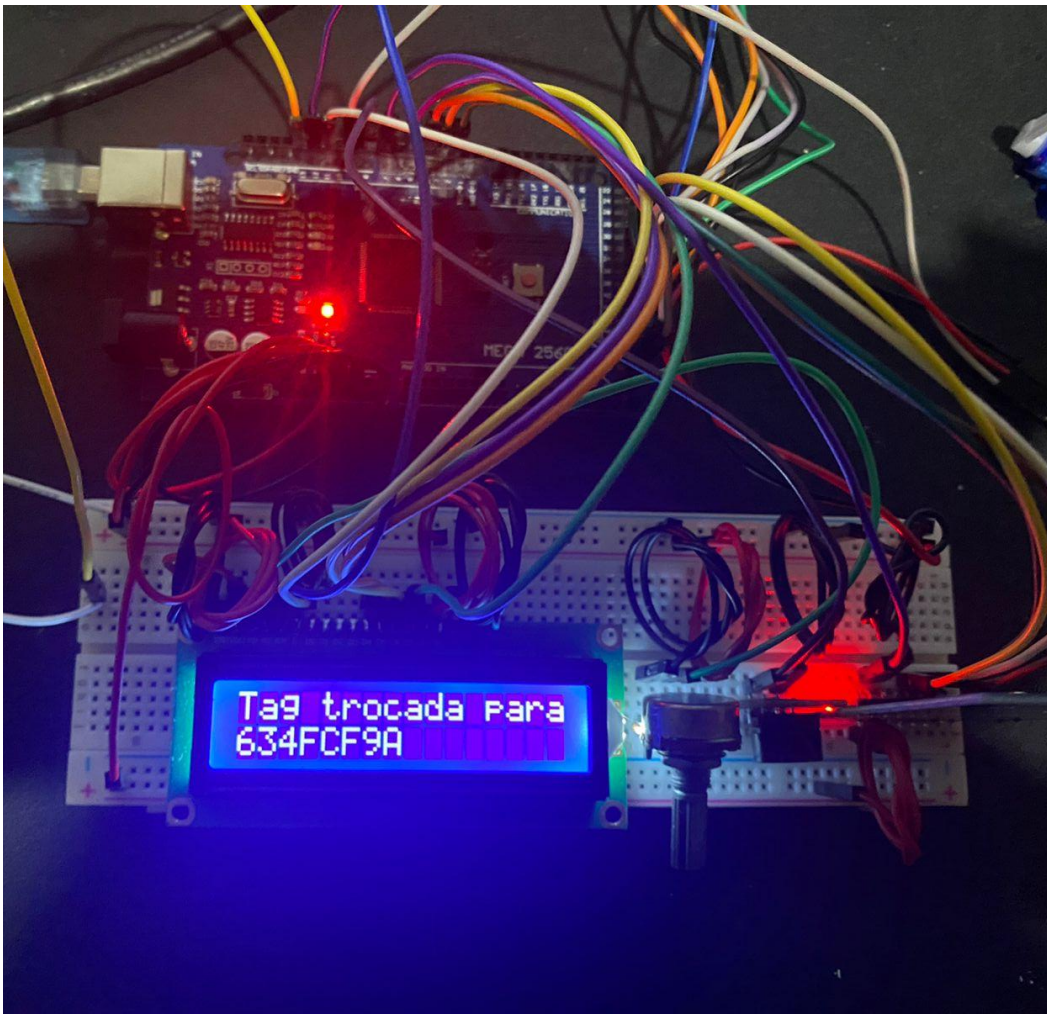
Figura 11 - Solicita aproximação de nova tag



Fonte: Autoria Própria

Ao aproximar o chaveiro RFID, a troca é confirmada e consolidada no sistema.

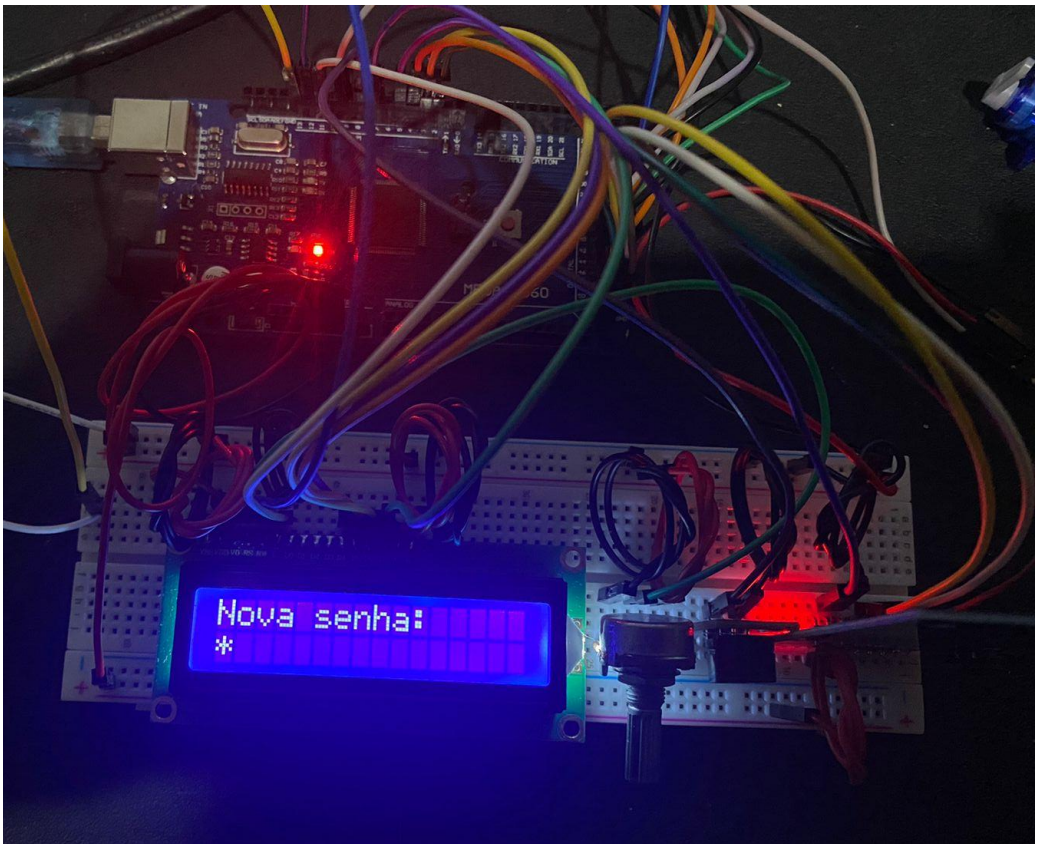
Figura 12 - Tag alterada



Fonte: Autoria Própria.

Optando pela alteração da senha, o sistema requisita a entrada da nova senha desejada.

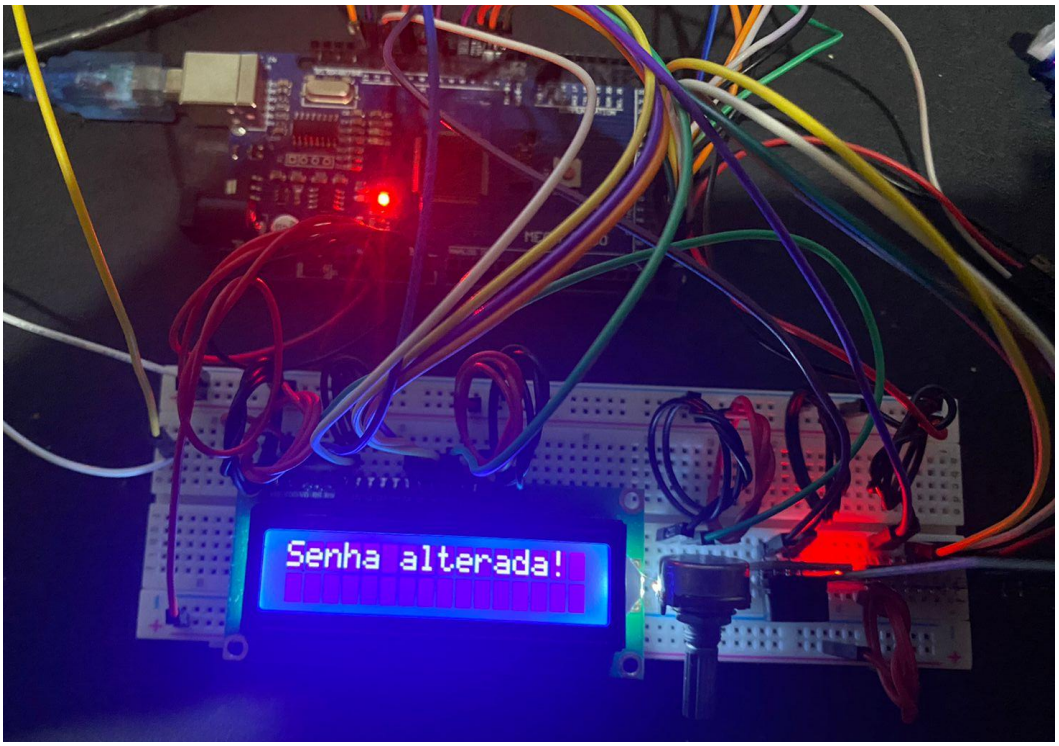
Figura 13 - Aquisição da nova senha.



Fonte: Autoria Própria

Após a inserção da nova senha desejada, o usuário deve concluir a digitação seguida da tecla '#' para confirmar a sequência digitada até o momento.

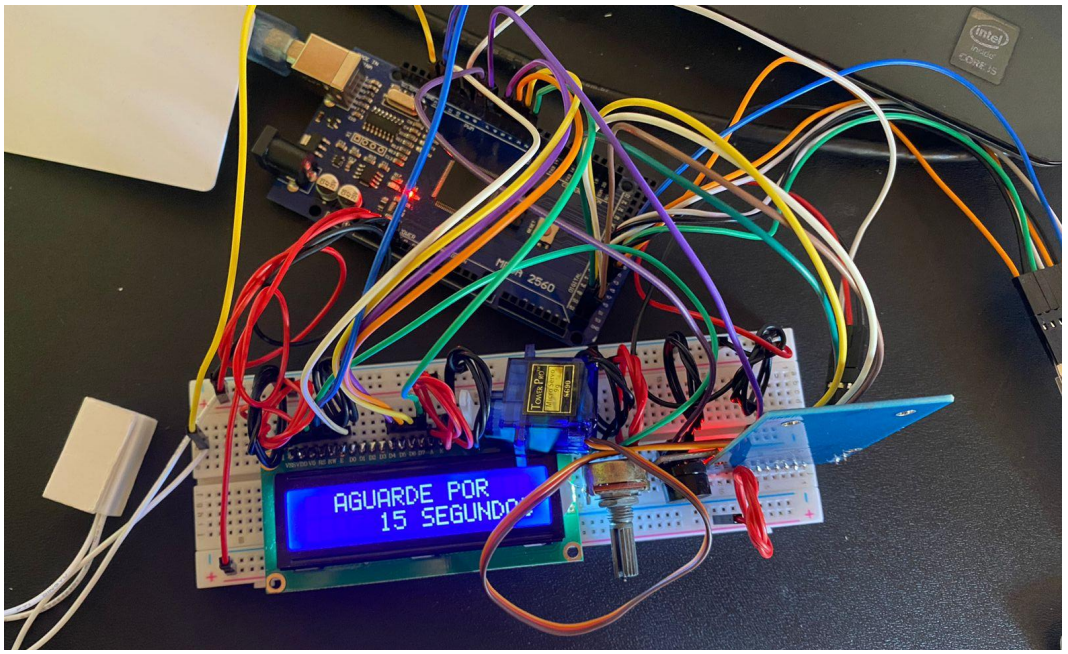
Figura 14 - Senha alterada.



Fonte: Autoria Própria

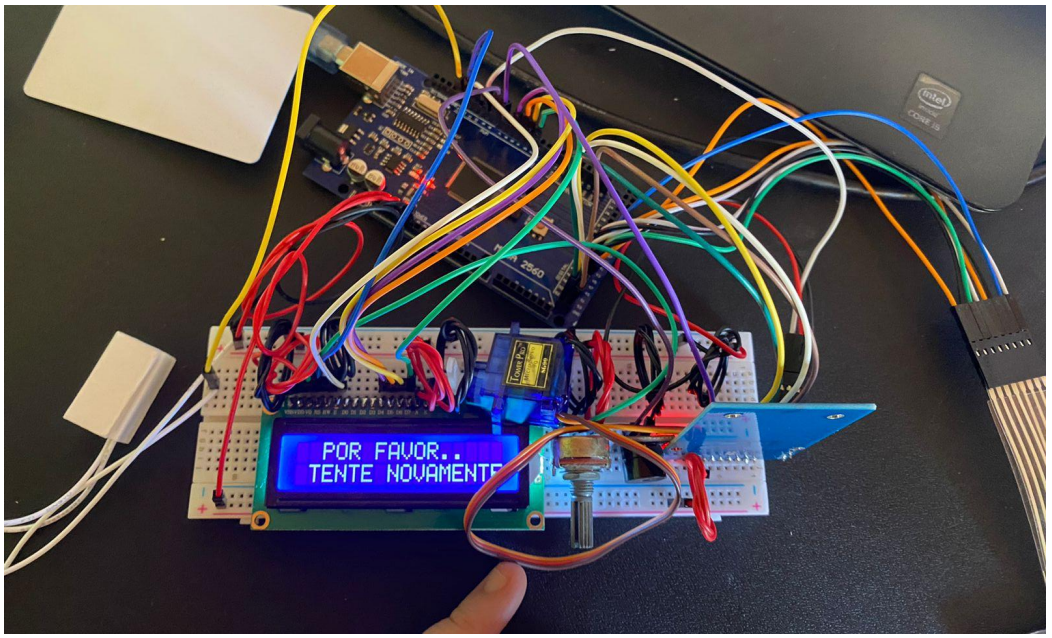
Quando o usuário falha na digitação da senha por quatro vezes consecutivas, o sistema ativa a função ‘tortura1()’, que é necessário aguardar um intervalo de 15 segundos para uma nova tentativa.

Figura 15 - Tortura 1



Fonte: Autoria Própria

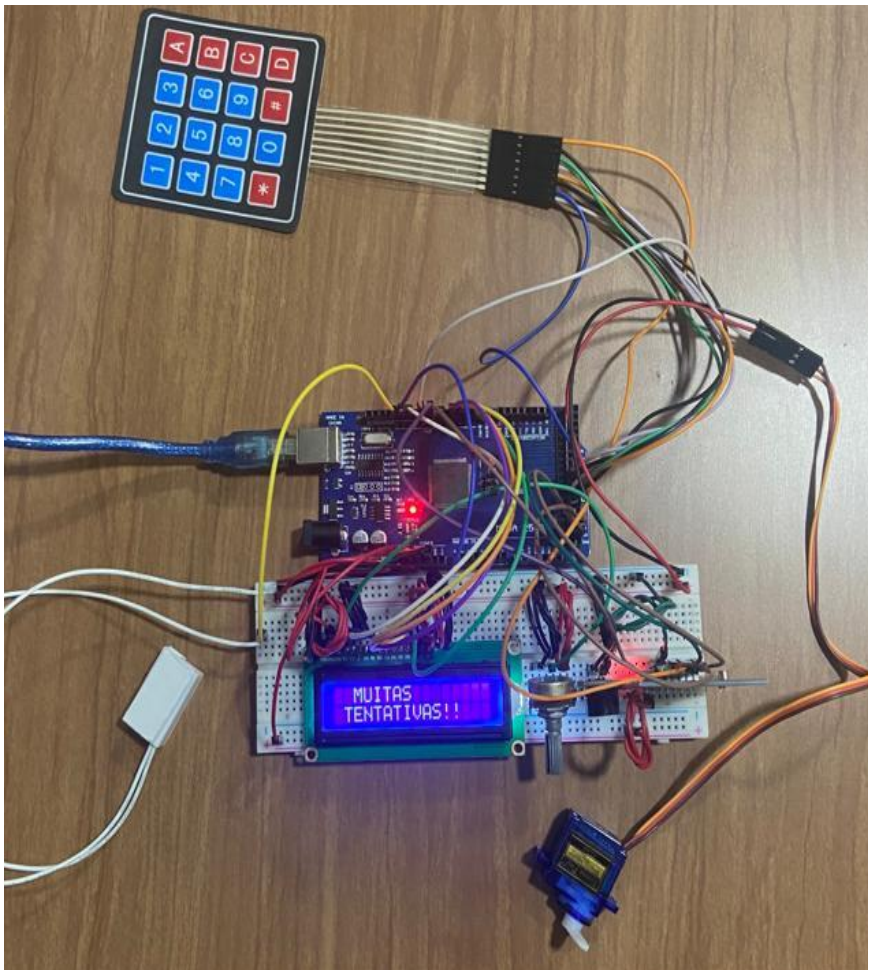
Figura 16 - Solicita nova tentativa.



Fonte: Autoria Própria.

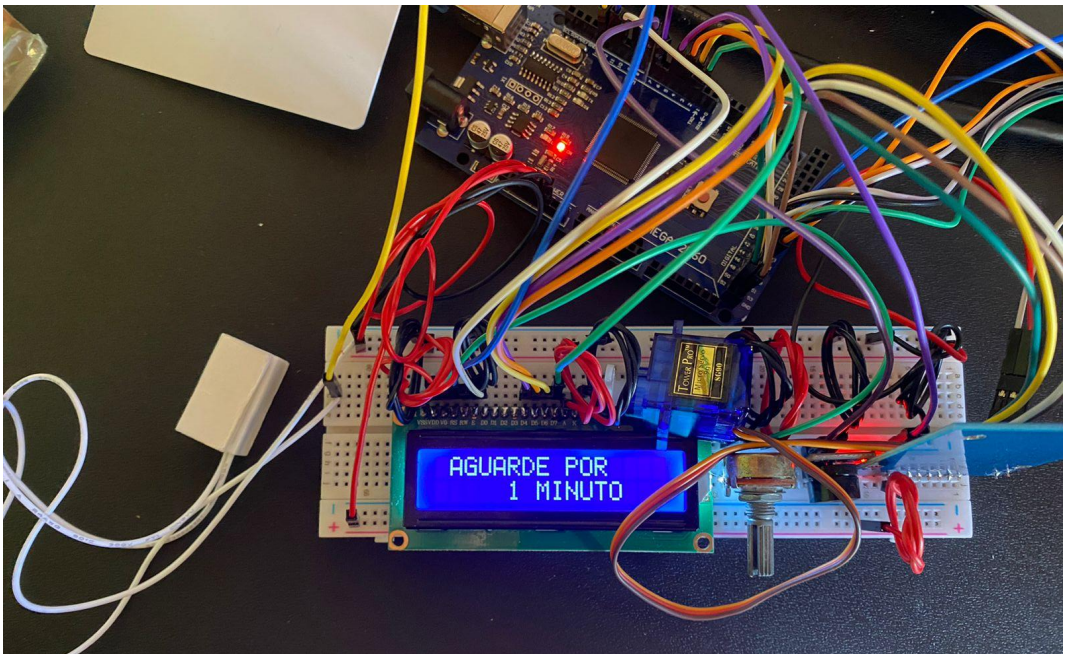
Caso o usuário cometa três erros consecutivos durante a inserção da senha após o intervalo de 15 segundos o sistema inicia a função ‘tortura2()’, no qual a penalidade é de 1 minuto de espera para uma nova tentativa, como mostra as figuras 16 e 17.

Figura 16 - Muitas tentativas erradas.



Fonte: Autoria Própria.

Figura 17 - Tortura 2.



Fonte: Autoria Própria.

5. CONSIDERAÇÕES FINAIS

O desenvolvimento deste projeto demonstrou a implementação de um sistema de segurança baseado em Arduino para simular o controle de acesso a um cofre. A integração de dispositivos como teclado matricial, LCD, servo motor, módulo RFID e sensor magnético permitiu a criação de um sistema robusto e versátil.

Durante a implementação, observou-se a importância da integração eficiente entre hardware e software. A utilização do teclado para inserção de senha e a inclusão posterior do módulo RFID ampliou as opções de autenticação, aumentando a flexibilidade do sistema.

A interação do usuário com o LCD forneceu um feedback claro das operações realizadas pelo sistema, garantindo uma melhor experiência do usuário e possibilitando mensagens informativas em cada etapa do processo.

O uso do servo motor para simular o mecanismo de abertura e fechamento da porta do cofre foi crucial para demonstrar visualmente o status de acesso. A adição do sensor magnético complementou essa simulação, indicando o estado aberto ou fechado da porta.

Além disso, as funções auxiliares implementadas no código foram essenciais para modular e organizar as diversas tarefas do sistema, tornando o código mais legível e fácil de manter.

Em síntese, este projeto demonstrou a aplicação prática de conceitos de segurança e controle de acesso em um ambiente controlado, fornecendo uma base sólida para futuras iterações e desenvolvimentos em sistemas de segurança baseados em Arduino.

6. REFERÊNCIAS BIBLIOGRÁFICAS

BAUERMEISTER, Giovanni. Como usar Servo Motor com Arduino. Disponível em: <https://blog.fazedores.com/como-usar-servo-motor-com-arduino/> Acesso em 29 de novembro de 2023.

Equipe MakerHero. Como utilizar o Display LCD 16×2 no Arduino? Disponível em:<https://www.makerhero.com/blog/como-utilizar-o-display-lcd-16x2/>
Acesso em 30 de novembro de 2023.

CASTRO, Giovanni de; CASSIOLI, Matheus. Usando o Teclado Matricial com Arduino. Disponível em:<https://www.robocore.net/tutoriais/usando-teclado-matricial-com-arduino> Acesso em 06 de dezembro de 2023.

STRAUB, Matheus Gebert. Projeto Arduino Controle de Acesso RFID. 2017. Disponível em: <https://www.usinainfo.com.br/blog/projeto-arduino-controle-de-acesso-rfid/>. Acesso em 06 de dezembro de 2023.

BABOS, Flávio. Módulo RFID: Como Usar No Arduino? [Controle De Acesso]. 2023. Disponível em: <https://flaviobabos.com.br/rfid-arduino/>. Acesso em 06 de dezembro de 2023.

Anexo I - Código do Projeto

```C/C++
#define NOTE_B0 31
#define NOTE_C1 33
#define NOTE_CS1 35
#define NOTE_D1 37
#define NOTE_DS1 39
#define NOTE_E1 41
#define NOTE_F1 44
#define NOTE_FS1 46
#define NOTE_G1 49
#define NOTE_GS1 52
#define NOTE_A1 55
#define NOTE_AS1 58
#define NOTE_B1 62
#define NOTE_C2 65
#define NOTE_CS2 69
#define NOTE_D2 73
#define NOTE_DS2 78
#define NOTE_E2 82
#define NOTE_F2 87
#define NOTE_FS2 93
#define NOTE_G2 98
#define NOTE_GS2 104
#define NOTE_A2 110
#define NOTE_AS2 117
#define NOTE_B2 123
```

```
#define NOTE_C3 131
#define NOTE_CS3 139
#define NOTE_D3 147
#define NOTE_DS3 156
#define NOTE_E3 165
#define NOTE_F3 175
#define NOTE_FS3 185
#define NOTE_G3 196
#define NOTE_GS3 208
#define NOTE_A3 220
#define NOTE_AS3 233
#define NOTE_B3 247
#define NOTE_C4 262
#define NOTE_CS4 277
#define NOTE_D4 294
#define NOTE_DS4 311
#define NOTE_E4 330
#define NOTE_F4 349
#define NOTE_FS4 370
#define NOTE_G4 392
#define NOTE_GS4 415
#define NOTE_A4 440
#define NOTE_AS4 466
#define NOTE_B4 494
#define NOTE_C5 523
#define NOTE_CS5 554
#define NOTE_D5 587
#define NOTE_DS5 622
#define NOTE_E5 659
#define NOTE_F5 698
#define NOTE_FS5 740
#define NOTE_G5 784
#define NOTE_GS5 831
```

```
#define NOTE_A5 880
#define NOTE_AS5 932
#define NOTE_B5 988
#define NOTE_C6 1047
#define NOTE_CS6 1109
#define NOTE_D6 1175
#define NOTE_DS6 1245
#define NOTE_E6 1319
#define NOTE_F6 1397
#define NOTE_FS6 1480
#define NOTE_G6 1568
#define NOTE_GS6 1661
#define NOTE_A6 1760
#define NOTE_AS6 1865
#define NOTE_B6 1976
#define NOTE_C7 2093
#define NOTE_CS7 2217
#define NOTE_D7 2349
#define NOTE_DS7 2489
#define NOTE_E7 2637
#define NOTE_F7 2794
#define NOTE_FS7 2960
#define NOTE_G7 3136
#define NOTE_GS7 3322
#define NOTE_A7 3520
#define NOTE_AS7 3729
#define NOTE_B7 3951
#define NOTE_C8 4186
#define NOTE_CS8 4435
#define NOTE_D8 4699
#define NOTE_DS8 4978
```

```
// Bibliotecas necessárias
#include <Keypad.h>
#include <LiquidCrystal.h>
#include <Servo.h>
#include <SPI.h>
#include <MFRC522.h>

// Configuração do módulo RFID
#define RST_PIN 7
#define SS_PIN 53
MFRC522 mfrc522(SS_PIN, RST_PIN);

// Configuração do sensor magnético da tranca
const int SENSOR_PIN = 13;
const int DARK_THRESHOLD = 100;

// Configuração do servo motor
Servo myservo;
const int redled = 10;
const int greenled = 11;
const int buzz = 8;
int pos = 0;

// Configuração do LCD
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);
const byte rows = 4;
const byte cols = 3;

// Configuração do teclado
const byte qtdLinhas = 4;
const byte qtdColunas = 4;
char matriz_teclas[qtdLinhas][qtdColunas] = {
 {'1', '2', '3', 'A'},
```

```
 {'4', '5', '6', 'B'},
 {'7', '8', '9', 'C'},
 {'*', '0', '#', 'D'}
};
byte PinosqtdLinhas[qtdLinhas] = {38, 39, 40, 41};
byte PinosqtdColunas[qtdColunas] = {42, 43, 44, 45};
Keypad meuteclado = Keypad(makeKeymap(matriz_teclas),
PinosqtdLinhas, PinosqtdColunas, qtdLinhas, qtdColunas);

// Configuração de senha aleatória
int tamanhoSenha = 4;
char password[10] = "4567";
char passwordEscrito[10];
int currentposition = 0;
int invalidcount = 0;
bool portaAberta = false;
byte tagEsperada[] = {0x52, 0x68, 0x50, 0x1B};

// Protótipos das funções
void displayscreen();
void keypress();
void incorrect();
void torture1();
void torture2();
void counterbeep();
void unlockbuzz();

// Função de inicialização
void setup() {
 displayscreen(); // Inicializa o LCD
 Serial.begin(9600); // Inicializa a comunicação serial
  pinMode(redled, OUTPUT); // Configura o LED vermelho como
saída
```

```
 pinMode(greenled, OUTPUT); // Configura o LED verde como saída
 pinMode(buzz, OUTPUT); // Configura o buzzer como saída
 myservo.attach(9); // Anexa o servo motor
  pinMode(SENSOR_PIN, INPUT_PULLUP); // Configura o pino do sensor magnético

 lcd.begin(16, 2); // Inicializa o LCD

 while (!Serial);      // Aguarda a inicialização da porta serial
 SPI.begin();    // Inicializa o barramento SPI
 mfrc522.PCD_Init();  // Inicializa o módulo RFID
 delay(4);       // Atraso opcional após a inicialização
  mfrc522.PCD_DumpVersionToSerial();   // Exibe detalhes do módulo RFID
  Serial.println(F("Scan PICC to see UID, SAK, type, and data blocks..."));// Mensagem para escanear PICC

 inicialTune(); // Toca uma melodia inicial
}

bool modoAdmin = false;
int estado = 0;

// Função principal

void loop() {

 if (modoAdmin) {
     estado = 0;
     menu(); // Exibe o menu
     switch (estado) {
     case 1:
```

```
        keypress(); // Aguarda a entrada do usuário
        trocarTag(); // Função para trocar a tag
        break;
        case 2:
        keypress(); // Aguarda a entrada do usuário
        trocarSenha(); // Função para trocar a senha
        break;
        }
        if (portaAberta) {
        lcd.clear();
        lcd.setCursor(0, 0);
        lcd.println(" ");
        lcd.setCursor(1, 0);
        lcd.print("Autorizado");
        lcd.setCursor(4, 1);
        lcd.print("BEM-VINDO");
        }
    } else {
        if           (mfrc522.PICC_IsNewCardPresent()           &&
mfrc522.PICC_ReadCardSerial()) {
        // Mostra os UID da tag
        Serial.print("UID da Tag:");
        for (byte i = 0; i < mfrc522.uid.size; i++) {
        Serial.print(mfrc522.uid.uidByte[i] < 0x10 ? " 0" : " ");
        Serial.print(mfrc522.uid.uidByte[i], HEX);
        }
        Serial.println();

        // Comparação do UID com o valor esperado
        if (verificarTag()) {
        Serial.println("Tag correta!");
        keypress(); // Aguarda a entrada do usuário
        modoAdmin = true;
```

```
        } else {
        Serial.println("Tag incorreta!");
        }

        delay(1000); // Evita leituras repetidas muito rápidas
        }
        if (portaAberta) {
        int proximidade = digitalRead(SENSOR_PIN);
        if (proximidade == LOW) {
        Serial.println("FECHADO");
        counterbeep(); // Emite um som indicando a proximidade
        inicialTune(); // Toca a melodia inicial
        }
        } else {
        char tecla_pressionada = meuteclado.getKey(); //  Verifica
se alguma tecla foi pressionada

        if (tecla_pressionada) { // Se alguma tecla foi pressionada
        Serial.print("Tecla pressionada : ");
        Serial.println(tecla_pressionada);
        passwordEscrito[currentposition] = tecla_pressionada;
        }

        if (currentposition == 0) {
        displayscreen(); // Atualiza a tela
        }

        int l ;

        char code = tecla_pressionada;
        if (code != NO_KEY) {
        lcd.clear();
```

```
lcd.setCursor(0, 0);
lcd.print("SENHA:");
lcd.setCursor(3, 1);
for (l = 0; l <= currentposition; ++l) {
lcd.print("*");
keypress(); // Aguarda a entrada do usuário
}
currentposition = currentposition + 1;

if (currentposition == tamanhoSenha) {
bool verificaSenha = true;
currentposition = 0;
for (l = 0; l < tamanhoSenha; ++l) {
if (passwordEscrito[l] != password[l]) {
      // Erro
      Serial.print("ERRO SENHA: ");
      Serial.print(passwordEscrito[l]);
      Serial.print(" != ");
      Serial.println(password[l]);
      verificaSenha = false;
}
}

if (verificaSenha) {
unlockdoor(); // Abre a porta
Serial.println("INVALIDO CONTADOR OK");
Serial.println(invalidcount);
invalidcount = 0;
} else {
invalidcount = invalidcount + 1;
Serial.println("INVALIDO CONTADOR BAD");
Serial.println(invalidcount);
incorrect(); // Feedback de senha incorreta
```

```
        }
        }

        if (invalidcount == 4) {
        ++invalidcount;
        torture1(); // Sequência de tortura 1
        }
        if (invalidcount >= 8) {
        torture2(); // Sequência de tortura 2
        invalidcount=5;
        }
        }
        }
 }


}

// Função para exibir o menu de escolha
void menu() {
 lcd.clear();
 lcd.print("Escolha:");
 lcd.setCursor(0, 2);
 lcd.print("1: Tag 2: Senha");

 char opcao = 0;

 while (opcao != '1' && opcao != '2') {
    opcao = meuteclado.getKey();
 }

 lcd.clear();
```

```
 switch (opcao) {
     case '1':
     lcd.print("Trocar Tag");
     estado = 1;
     delay(2000);  // Aguarda por 2 segundos para exibir a
mensagem
     break;
     case '2':
     lcd.print("Trocar Senha");
     estado = 2;
     // Chame a função para trocar a senha aqui
     delay(2000);  // Aguarda por 2 segundos para exibir a
mensagem
     break;
 }

 lcd.clear();
}

// Função para trocar a senha
void trocarSenha() {
 lcd.clear();
 lcd.print("Nova senha:");
 lcd.setCursor(0, 1);

 byte index = 0;

 while (true) {
     char tecla = meuteclado.getKey();
     if (tecla) {
     keypress();

     if (tecla == '#' and index > 0) {
```

```
    password[index] = '\0'; // Adiciona o caractere nulo para
indicar o final da string
    tamanhoSenha = index;
    break; // Finaliza a entrada da senha
    }

    if (index < 10) {
    password[index++] = tecla;
    if (index == 10){
    tamanhoSenha = index;
    break;
    }
    lcd.print('*'); //  Máscara  para  esconder  os  caracteres
digitados
    }
    }
 }

 Serial.println(); // Nova linha para melhorar a legibilidade
 lcd.clear();
 lcd.print("Senha alterada!");
 delay(1000);
 invalidcount=0;
 Serial.print("Senha alterada para: ");
 Serial.println(password);
 modoAdmin = false;
}

// Função para trocar a tag RFID
void trocarTag() {
 lcd.clear();
 lcd.print("Aproxime cartao:");
```

```
 while (!mfrc522.PICC_IsNewCardPresent()) {
    delay(1000);
 }

 if (mfrc522.PICC_ReadCardSerial()) {
    for (byte i = 0; i < mfrc522.uid.size && i < sizeof(tagEsperada); i++) {
    tagEsperada[i] = mfrc522.uid.uidByte[i];
    }

    Serial.print("Tag trocada para:");


    lcd.clear();
    lcd.print("Tag trocada para:");
    lcd.setCursor(0, 1);

    for (byte i = 0; i < sizeof(tagEsperada); i++) {
    lcd.print(tagEsperada[i], HEX);
    Serial.println(tagEsperada[i], HEX);
    }

    delay(2000);  // Aguarda por 2 segundos para exibir a mensagem
    modoAdmin = false;
 }

 mfrc522.PICC_HaltA();
 mfrc522.PCD_StopCrypto1();
}

// Função para verificar se a tag RFID é válida
bool verificarTag() {
```

```
 for (byte i = 0; i < mfrc522.uid.size; i++) {
     if (mfrc522.uid.uidByte[i] != tagEsperada[i]) {
     return false; // UID não corresponde
     }
 }
 return true; // UID corresponde
}

// Função para tocar uma melodia inicial
void inicialTune() {
  int melody[] = {NOTE_E5, NOTE_E5, NOTE_E5, NOTE_B4, NOTE_D5,
NOTE_C5, NOTE_C5, NOTE_D5, NOTE_E5, NOTE_E5, NOTE_D5, NOTE_D5};
 int noteDurations[] = {8, 8, 4, 8, 8, 4, 8, 8, 8, 8, 8, 4};

 for (int i = 0; i < sizeof(melody) / sizeof(melody[0]); i++) {
     int noteDuration = 1000 / noteDurations[i];
     tone(buzz, melody[i], noteDuration);
     int pauseBetweenNotes = noteDuration * 1.30;
     delay(pauseBetweenNotes);
     noTone(buzz);
 }
}

// Função para abrir a porta
void unlockdoor() {
 delay(900);

 lcd.setCursor(0, 0);
 lcd.println(" ");
 lcd.setCursor(1, 0);
 lcd.print("Autorizado");
 lcd.setCursor(4, 1);
 lcd.print("BEM-VINDO");
```

```
 unlockbuzz();

 for (pos = 90; pos >= 0; pos -= 5) {
     myservo.write(pos);
     delay(5);
 }
 delay(2000);

 portaAberta = true;
}

// Função para lidar com código incorreto
void incorrect() {
 delay(500);
 lcd.clear();
 lcd.setCursor(1, 0);
 lcd.print("CODIGO");
 lcd.setCursor(1, 1);
 lcd.print("INCORRETO");
 lcd.setCursor(15, 1);

 Serial.println("CÓDIGO INCORRETO. NÃO AUTORIZADO");
 digitalWrite(redled, HIGH);
 tone(buzz, NOTE_E1, 1000);
 delay(500);
 noTone(buzz);
 delay(3000);
 lcd.clear();
 digitalWrite(redled, LOW);
 displayscreen();
}
```

```
// Função para limpar a tela do LCD
void clearscreen() {
 lcd.setCursor(0, 0);
 lcd.println(" ");
 lcd.setCursor(0, 1);
 lcd.println(" ");
 lcd.setCursor(0, 2);
 lcd.println(" ");
 lcd.setCursor(0, 3);
 lcd.println(" ");
}

// Função para emitir um som de tecla pressionada
void keypress() {
 tone(buzz, NOTE_C6, 100);
 delay(50);
 noTone(buzz);
}

// Função para exibir a tela inicial no LCD
void displayscreen() {
 lcd.clear();
 lcd.setCursor(0, 0);
 lcd.println("*DIGITA A SENHA*");
}

// Função para posicionar o servo
void armservo() {
 for (pos = 180; pos <= 180; pos += 50) {
    myservo.write(pos);
    delay(5);
 }
 delay(5000);
```

```
 for (pos = 180; pos >= 0; pos -= 50) {
     myservo.write(pos);
 }
}

// Função para emitir um som de desbloqueio
void unlockbuzz() {
 for (int i = 0; i < 4; i++) {
     tone(buzz, NOTE_C6, 100);
     delay(500);
     noTone(buzz);
 }
}

// Função para contar beeps
void counterbeep() {
 delay(1200);

 lcd.clear();

 lcd.setCursor(2, 0);
 delay(200);
 lcd.print("FECHANDO EM:");
 for (int i = 5; i > 0; i--) {
     lcd.setCursor(7, 1);
     lcd.print("0");
     lcd.print(i);
     keypress();
     delay(1000);
 }

 for (pos = 0; pos <= 90; pos += 5) {
```

```
    myservo.write(pos);
  }
  delay(15);

  currentposition = 0;

  lcd.clear();
  displayscreen();

  lcd.clear();
  lcd.setCursor(4, 0);
  lcd.print("FECHADO!");
  delay(440);

  portaAberta = false;
}

// Função de tortura 1
void torture1() {
  lcd.clear();
  lcd.setCursor(2, 0);
  lcd.print("AGUARDE POR ");
  lcd.setCursor(5, 1);
  lcd.print("15 SEGUNDOS.");
  digitalWrite(buzz, HIGH);
  delay(15000);
  digitalWrite(buzz, LOW);
  lcd.clear();
  lcd.setCursor(2, 0);
  lcd.print("POR FAVOR..");
  lcd.setCursor(1, 1);
  lcd.print("TENTE NOVAMENTE");
  delay(3500);
```

```
 lcd.clear();
}

// Função de tortura 2
void torture2() {
 lcd.clear();
 lcd.setCursor(1, 0);
 lcd.print(" ");
 lcd.setCursor(2, 0);
 lcd.print("MUITAS");
 lcd.setCursor(0, 1);
 lcd.print(" TENTATIVAS!! ");
 delay(1500);
 lcd.clear();
 lcd.setCursor(1, 0);
 lcd.print("AGUARDE POR ");
 lcd.setCursor(4, 1);
 lcd.print(" 1 MINUTO");
 digitalWrite(buzz, HIGH);
 delay(55000);
 lcd.clear();
 digitalWrite(buzz, LOW);
 lcd.setCursor(2, 0);
 lcd.print("TENTE ACERTAR");
 lcd.setCursor(1, 1);
 lcd.print("TA DIFICIL??");
 delay(2500);
 lcd.clear();
 lcd.setCursor(2, 0);
 lcd.print("Ha Ha Ha Ha");
 delay(1700);
 lcd.clear();
}
```